[redacted]

510.84
Il6r
no.339 Report No. 339

Math

ILLIAC IV

QUARTERLY PROGRESS REPORT

January, February, and March 1969

Contract No.
US AF 30(602)4144

ILLIAC IV Doc. No. 219

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

ILLIAC IV
QUARTERLY PROGRESS REPORT
January, February, and March 1969

Contract No.
US AF 30(602)4144

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois
61801

May 1, 1969

This work was supported in part by the Department of Computer Science, University of Illinois, Urbana, Illinois, and in part by the Advanced Research Projects Agency as administered by the Rome Air Development Center, under Contract No. US AF 30(602)4144.

TABLE OF CONTENTS

REPORT SUMMARY

Introduction

Several major problem areas were reported during the last quarter. A detailed report on the problems and the solutions was presented at the ILLIAC IV Advisory Committee Meeting in Washington, D.C. on February 26, 1969. As a result of the progress made and the solution to the problems encountered by ILLIAC IV, the Advisory Committee recommended unanimously that the Project continue to completion and gave their approval of the proposed solutions. The following are the major problem areas.

Semiconductor Devices

ILLIAC IV is being implemented with the TI (Texas Instruments) ECL dual-in-lines (DIL's). The PE has been redesigned to eliminate the MSI devices and the redesign is completed. The mechanical configuration for the PE now resembles the breadboard built by TI for Burroughs. The PE boards are approximately 3 1/2" x 4 1/2" and contain up to 20 DIL's. Artwork masters for printed circuit board manufacturing are expected to be completed in the next quarter. The Control Unit required no major change in design, with the exception that the clock had to be increased from 40 nsec to 50 nsec.

Memories Systems

Burroughs is in the process of signing a contract with Fairchild to provide memories for the ILLIAC IV PE's. These memories will be semiconductor memories with 188 nsec cycle time. Burroughs should receive the first memories in December, 1969.

Impact on Cost, Schedule, and Performance

ILLIAC IV's performance is essentially the same with the DIL technology as was proposed with the MSI devices. The table below indicates the original performance and the projected performance:

Effect of Change of Semiconductor Implementation
Technique and Clock Rate

	Original Target	Present Outlook
Additions/Second	10^9 per Second	$\frac{\text{PU's}}{250} \times 10^9$ per Second
Multiply Time	400 Nanoseconds	450 Nanoseconds
Memory Cycle Time	250 Nanoseconds	200-250 Nanoseconds
Disk File Transfer Rate	10^9 Bits per Second	10^9 Bits per Second
Percent Time PE Busy	$> 91\%$	$> 91\%$

The projected cost to complete for a one (1) quadrant system is in the order of $26,000,000. The present schedule cost for installation of the system is September, 1970. Due to the increase in cost on the ILLIAC IV project, it has been decided to install a one (1) quadrant system at the University rather than a four (4) quadrant system.

Project Review

The University has reached an agreement with Burroughs to review project status on a monthly basis. Personnel from the University of Illinois will visit Burroughs each month and review progress and budgets on a task-by-task basis. The result of these reviews will be incorporated into the monthly status report produced by Burroughs. These reviews should provide the University with the information necessary to determine if Burroughs is meeting the schedule and cost objectives presented to the Advisory Committee.

1. HARDWARE

1.1 Simulation and Debugging of PE Logic

1.1.1 PE Logic Simulator

The PE logic simulator has been modified so that it can accept up to 47 independent test cases at the same time and simulate them in parallel. The previous format of the input file is still valid. If the number, n, of operands is less than 47, the simulator prints out n sets of specified contents. An estimated execution time per clock per operand is in the order of .1 second.

1.1.2 Equation Generator Program

Due to some errors in input data to the PE equation generator program, this program had to be run again on the corrected input data of the control logic. Because the program execution took a long time, the programming has been improved.

A program for simplifying Boolean equations for the PE control logic has been written for the convenience of logical designers and diagnosticians. This program has the capability to simplify equations using DeMorgan's Law and a few other propositions of Boolean algebra.

1.1.3 Debugging of the PE Design

The simulator has been used extensively during this period in debugging the logic design. The third approach mentioned in the last Progress Report has been used in preparing test routines and approximately 200 errors have been found and corrected. The implementation of most of the arithmetic instructions with variants has been corrected. In the next quarter the use of the simulator will be mainly for verification, rather than correction, of design.

1.2 PE Diagnostics Generation

1.2.1 Assembled Code Translator (ACT)

The Assembled Code Translator program is in the process of being expanded to provide output data in a parallel form to the PE simulator. The revision involves searching a file created by ACT to determine the number of clocks in the instruction that the simulator is to execute. The F-signals for each clock in the instruction and the data associated with each clock for up to 47 executions of the instruction are then written on the output file. If the instruction is to be executed for more than 47 sets of data, it is necessary to repeat the F-signals with the additional data since the PE simulator can handle a maximum of 47 inputs. At the present time ACT II, the revised version of ACT, is being debugged.

1.2.2 Combinational Tests

Generation of new test patterns for combinational logic has been delayed until the next period because of the urgency of other tasks.

1.2.3 Path Tests

The path test development is in its final step. The main task of this quarter was the completion of the path test program. The insertion of initialization microsequences was done. The set of path tests and ordering sequences are fed into a program called TOP/FORM which produces a file called TOP/PEXIN containing the ordered sequence path tests. The TOP/PEXIN will be fed into the PE Exerciser Test Assembly Program which will produce an object code of the path test program and a fault dictionary.

The path test generator program (including the expected response calculator program) was maintained and revised during this period. A short routine for checking the clearing function of the main registers has been written and attached to the top of the path test program.

1.2.4 Control Logic Tests

An effort of generating test cases for the PE control logic has been initiated in this period. Some discussions have been concentrated on the automatic generation of the test cases. A translation from the logic equations of the PE control logic to the inputs to the PGM (Path Generating Method) program is being investigated for possible implementation in the next period.

1.3 CU Logic Simulator System

1.3.1 General

The majority of the programs comprising the CU Card Board Logic Simulator System have been coded, and debugging of the program is proceeding. A number of deficiencies (e.g., missing wiring, spurious records) in the netlists provided by Burroughs for experimental and debugging use have been discovered and corrected versions have been promised. It should be possible to generate and test a complete simulator by the time these corrections are received.

Work has progressed slowly on the CU Section Logic Simulator System, primarily due to a lack of programming manpower.

1.3.2 CU Card Simulator Generator

The initial version of the CU Card Simulator Generator system was written this quarter in accordance with the specifications developed during the previous quarter. The simulator generator system consists of seven subprograms which perform the following tasks:

1) netlist reformatting and updating,
2) signal name sort and simulation variable identifier assignment,
3) pin sort and directed graph generation,
4) initial level assignment,
5) loop detection,
6) final level assignment, and
7) simulator body generation.

Although the algorithms used in this system are almost identical to those used in the PE logic simulator generator, the programs have been rewritten to provide system refinements and increased efficiency. The major features are listed below.

1) Since there are about one hundred CU cards, a processing history is automatically generated for every set of files produced by any program. Further, file identifiers are automatically assigned to include the board name and a program run number.

2) The entire simulator generation process is oriented towards producing an efficient simulator capable of simulating many test cases in parallel.

3) Simulator efficiency is increased by using in-line logic equation coding, rather than procedure calls as in the PE simulator.

4) A netlist updating facility has been provided. Minor changes to the netlist may easily be incorporated by preparing update cards and rerunning the simulator generator system.

Typical execution times on the B5500 for the complete CU card simulator generation process are:

	TOFUND (850 records)	TIFUNA (1350 records)
1) UPDATE	45 sec	1 min 15 sec
2) WEDTR 1*	2 min 50 sec	4 min
3) WEDTR 2	40 sec	1 min
4) LEVELER	30 sec	1 min
5) REDUCE	4 sec	2 sec
6) HSEKEEP	3 sec	5 sec
7) SIMGEN	30 sec	40 sec
Totals	5 min 22 sec	8 min 2 sec

*This program currently uses a relatively inefficient linked-list bubble sort. It will soon be modified to use the highly efficient ALGOL SORT verb; anticipated running times are on the order of those for WEDTR 2.

1.3.3 CU Card Simulator

The card simulator "head" and "tail" have been completed and debugged. These are sections of code which are identical to each simulator. They contain such things as declarations, loop control, and a quite large simulator initialization and input control block. Preliminary tests indicate that the chosen format for simulator input is very efficient, resulting in low programmatic overhead for simulator input-output.

1.3.4 Simulation Results Display Program

Rather than printing simulation results as they are generated, the simulator merely dumps the values of all simulation variables at each step to an output file for subsequent postprocessing by a results display program. This permits the simulator to operate at maximum speed by eliminating the necessity for it to stand essentially idle while time consuming I/O bound results' printouts take place. The values of the simulation variables, in a user-specified format, are printed by the separate results display program.

The coding for this program has been completed during the quarter, and debugging is almost complete. Initial design predictions have been borne out entirely. Through the use of ALGOL stream (character handling) procedures, a very small, fast program has been produced which runs almost totally in the shadow time of the necessary I/O transfers.

The simulator user may (through the TESLA or PEXTAP languages) specify printing of any subset of the simulation variables in any desired order. Printouts may be in octal or binary representation. Expected results may also be presented for display and comparison to the actual results, and printout may be suppressed in cases where these match.

1.3.5 TESLA

A complete BNF syntax of the new simulation test control language, TESLA, was developed during the quarter. This was transferred to punched cards and used as input to the ILLIAC IV Translator Writing System. After a few trivial modifications to compensate for some vagaries in the BNF to Floyd Production Language translation algorithm, a parser for TESLA was successfully produced.

Semantic action calls were added at appropriate points in the syntax; the semantic structure of the language was coded; and semantic action routine specifications were developed. Coding for all but the most difficult (and least useful) semantics have been completed and are currently being debugged.

The ILLIAC IV Translator Writing System has been the keystone in the rapid generation of the TESLA compiler. The necessity for several man-months of labor for development of a scanner, parser, and symbol-table building routines was obviated by TWS. Minor difficulties were experienced because of a current lack of documentation and instructions for generating a compiler using TWS and the relative inefficiency of the translation programs, but these may be ascribed to the yet developmental nature of the system and will undoubtedly be ultimately corrected.

A translator for generating simulator and results display input files from simple symbolic representations of the actual simulator input was also written. This has been used to provide dummy programs for simulator debugging while the TESLA compiler is being written.

1.3.6 CU Section Logic Simulator System

Because of a shortage of programming manpower, maximum effort has been devoted to the completion of the CU Card Logic Simulator System, and efficiency and program behavior in a multiprogramming computer environment. Radically different concepts such as a partially or even fully interpretive simulator are being examined, and it is even possible that two entirely different (but compatible) architectures will be selected, one for the daytime (multiprogramming) environment and the other for nighttime (exclusive-use-of-machine) environment.

1.4 CU Diagnostics Generation

1.4.1 CU Board Test Generator

The organization of the CU Board Test Generator System has been investigated during this period. The system will be composed of Failure Mode Collector, Failure Mode Selector, Pattern Generator, Bad Board Simulator, Good Board Simulator, and Test Generation Controller.

Failure modes of chips on the board are collected by the FM collector and organized in a set of failure modes of the board. The patterns are generated assuming a grouping of input signals to the board. The test generation controller requests a failure mode and a bit pattern from the pattern generator. The failure mode is put into the bad board simulator and its response is compared with that of a good board simulator (free of failure). If there is no possible detection, the controller looks for another pattern; if these patterns are exhausted, it requests another failure mode until completing the set in the case of a detection. The pattern and failure modes are recorded to produce a set of tests.

All the programs will be written in the next period except the simulators that will be created by the CU Board Simulator Generator.

1.4.2 CU On-Line Diagnostics

Because of the termination of the Self-Diagnostic Generation (SDAS) effort at Burroughs, Paoli, some of the tasks for CU on-line diagnostics were transferred to the University in this period.

The identification of programs required for on-line diagnostics has been started. An extensive effort in this area is expected in the next period.

1.5 Design Automation

The Post-Processor Program was written and returned to Burroughs. Some additions and modifications were necessary due to some small changes in the wiring rules. As the program is used in the production phase, the identification will continue of places where additions to the post-processor will relieve manual checking and will smooth the design flow for Control Unit's PC Boards.

The University has assumed the responsibility for the development of additional new software in the DA Development area. It is expected that this new software can be developed with undergraduate hourly programmers. In the past, hourly programmers have been sufficient to develop DA software.

The DA Production Programs (Router, Printer Plot, and Post-pro) are being run at the University on the third shift. During this quarter, the last of the operational difficulties in running the Burroughs programs on the University's B5500 seem to have been ironed out. The MDS 1103 Data Link has come up to full operational status, and data is flowing in both directions with no difficulties.

Work has started on a program called Logic Systems Designer's on Line Testor. This program will use the logic model built by the Delay Check Program and will act as an Information Retrieval Program to allow the Logic Designer to examine the logic nets after layout has been finished.

2. SOFTWARE

2.1 Operating System

The specification of the ILLIAC IV operating system was completed in this quarter and has been described in a report. The report describes those sections residing principally on the B6500 computer--the job parser, program collector, disk file allocator, data processor, execution monitor, job partners, and hardware supervisor--and the sections of the operating system resident in the ILLIAC IV - OS4 and the loader. A brief description of the ILLIAC IV hardware and discussions of file security and interactive uses of the ILLIAC IV are also included.

The operating system group is obligated to furnish a workable version of the operating system on or about 1 October 1969. To this end a schedule has been formulated. The tasks to be performed were sorted into four groups.

1) Those tasks which appear to require minimal research

   a) Job Parser
   b) Disk File Allocator
   c) Data Processor
   d) Execution Monitor

   Deadline: Completely designed, coded, and partially debugged by 15 June.

2) Those tasks requiring intimate knowledge and closely connected design

   a) Loader
   b) Program Collector
   c) OS4

   Deadline: Complete specification by 1 June. Coded by 15 August.

3) Those tasks requiring further design and/or research

   a) Hardware Supervisor
   b) Job Partner
   c) Execution Monitor

Deadline: The research has the same deadline as task 2. The current version of the job partner given in the Operating System Description will be coded. No firm deadlines have been set on this sub-task.

4) Those tasks relating to the implementation of the system

a) Debug - Simulator

Deadline: Feasibility and desirability must first be assessed. However, a preliminary version is to be completed by 15 June to be of use in debugging the sub-tasks of task 1.

Major reviews of the development of the system are scheduled in May and June. By mid-June the entire design will be fixed, and a sizable portion of the system will be in the coding stage or beyond.

2.2 Translator Writing System

Some small changes have been made to the BNF → FPL syntax preprocessing algorithm and to its implementation. These changes have improved the speed and efficiency of the syntax preprocessor. Further changes which will increase the class of grammars acceptable to the system and improve the automatic error recovery are presently being considered and/or implemented. A first stage of documentation of the syntax preprocessor has been completed (Department of Computer Science Report No. 304) and further documentation is continuing.

The generation of parser pseudo-orders was modified to implement "one fell swoop" tests whenever there is a list of different single symbols to be tested in either the lookahead or the main stack followed by the same action in each case. This is accomplished by setting bits in a packed Boolean array row in the preprocessor. In the parser, the symbol to be checked is used to see if its bit in the appropriate row is "on."

A TRANQUIL compiler has been implemented and is currently being used to improve the system.

The work on TWINKLE is proceeding. The syntax specification has been made and is constantly being improved, and semantic routines are being written.

A new precedence function has been added to TWST in order to express the notion of operator precedence in a more compact form than at present. Several other primitives have been introduced including one to compare two atoms of an input stream without the atoms concerned being reserve words. For example, the identifiers which begin and end a PL1 compound statement could be compared directly by using TWST syntax instead of semantics.

Extensive tests have been run on ISL translators generated by the TWS system. Since ISL was the first, reasonably large language completely implemented using the TWS system, more realistic speed measurements could be performed on it.

Three different kinds of parsers can be used in compilers generated using the TWS system.

1) The first type of parser is a "slow" interpretive parser with a good error recovery which is designated PARSER$\underline{\text{I}}$. This parser reads and executes a stream of coded orders called "parser instructions".

2) The second type of parser is a "fast" directly coded parser with good error recovery and is designated PARSER$\underline{\text{E}}$. In this version the actions performed are the same as in PARSERI but, instead of reading parser instructions, PARSERE contains such instructions directly coded in ALGOL.

3) A "fast" directly coded parser with fair error recovery which is designated PARSER$\underline{\text{F}}$ is the third type of parser. This is similar to PARSERE except that some error recovery facilities are dropped in order to gain some speed for production runs; PARSERE would be used for debugging runs.

For an ISL compiler, using as input a "typical" ISL program with 2000 cards, the results were:

Type of Parser	Speed in Cards per Minute of Processor Time
PARSERI	452
PARSERE	520
PARSERF	524
"brute-force" ISL translator	2056

These results suggest the following guidelines for further development of the system:

1) Since PARSERE and PARSERF run at practically the same speed, PARSERF could be dropped.

2) Although PARSERE runs somewhat faster (processor time) than PARSERI, the total time in the machine is longer since PARSERE yields an extremely large compiler; moreover, PARSERE cannot be multiprocessed while PARSERI multiprocesses very well. This points to PARSERI as the most promising type of parser and efforts are being initiated to improve it.

Finally, it could be noted that the "brute-force" ISL translator runs four times faster than the TWS generated compilers.

2.3 Compilers and Translators

2.3.1 TRANQUIL

During this quarter, work in the pass 2 semantics for the TRANQUIL compiler continued. Progress was slowed, in part, by the frequent unavailability of the B5500 and its increasingly heavy usage.

Work has been initiated on the development of a more generalized data mapping declaration which will permit skewing an array an arbitrary distance based on the array size. Work has also been started on the implementation of all set operations.

Procedures to handle efficient utilization of PE registers during the evaluation of arithmetic expressions have been completed.

Several minor modifications were made to the syntax of TRANQUIL. Included in these changes was the addition of a new set declaration, PATSET, for pattern sets. A pattern set is a multi-dimensional set that will be stored in ILLIAC IV as a mode pattern and may be used to access scattered array elements. The maximum range for the values of each dimension of the set may now be specified in a manner analogous to array subscript bound declarations. A lower bound of 1 is assumed if none is specified. All ALGOL coding for the declaration and definition of sets is now complete.

The storage allocation and array partitioning algorithms have also been slightly modified. They will now provide for array type data to be partitioned into blocks 128 x 128 words but stored completely within one quadrant of the ILLIAC IV. An array declared to be 150 x 150 would now be partitioned and arranged in PE memory (see Figures 1 and 2). The algorithms themselves are used only in the calculation of the PE number and row number for a particular array element. The block number computations remain unchanged.

2.3.2 GLYPNIR

The GLYPNIR compiler is now in the final stages of debugging. The full compiler for Version I is running, and the code generated by it is being checked and simulated. A user's manual for Version I is now being printed and should soon be released.

Work is also starting on Version II, which will include many extensions to the basic language as well as a faster compiler and better debugging facilities. The extensions, on which work has been started, include more flexible subroutine and control statements, as well as some basic I/O statements.

2.4 Graphics

A serial algorithm for doing shadowing by multiple point light sources of objects as well as half-tone and hidden plane consideration is near finalization. It also considers such things as direct reflection, transmission (transparent or semi-transparent objects), and color. No such algorithm yet exists for serial (or parallel) machines. Parallel

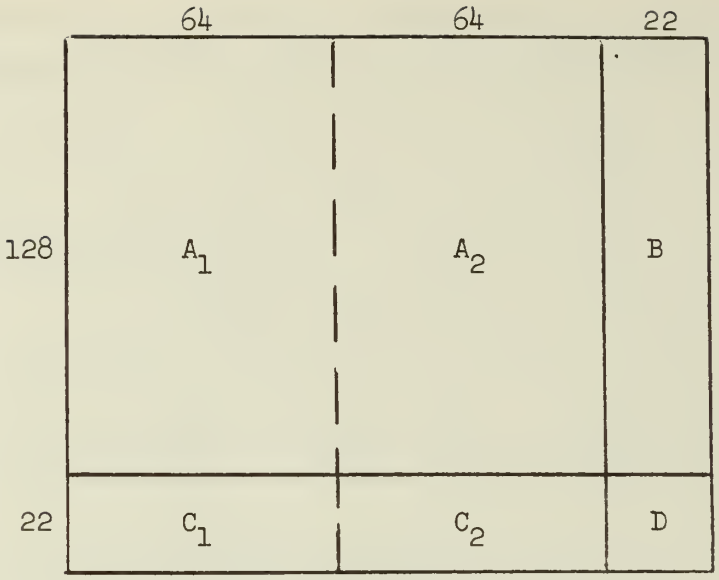

Figure 1. Array

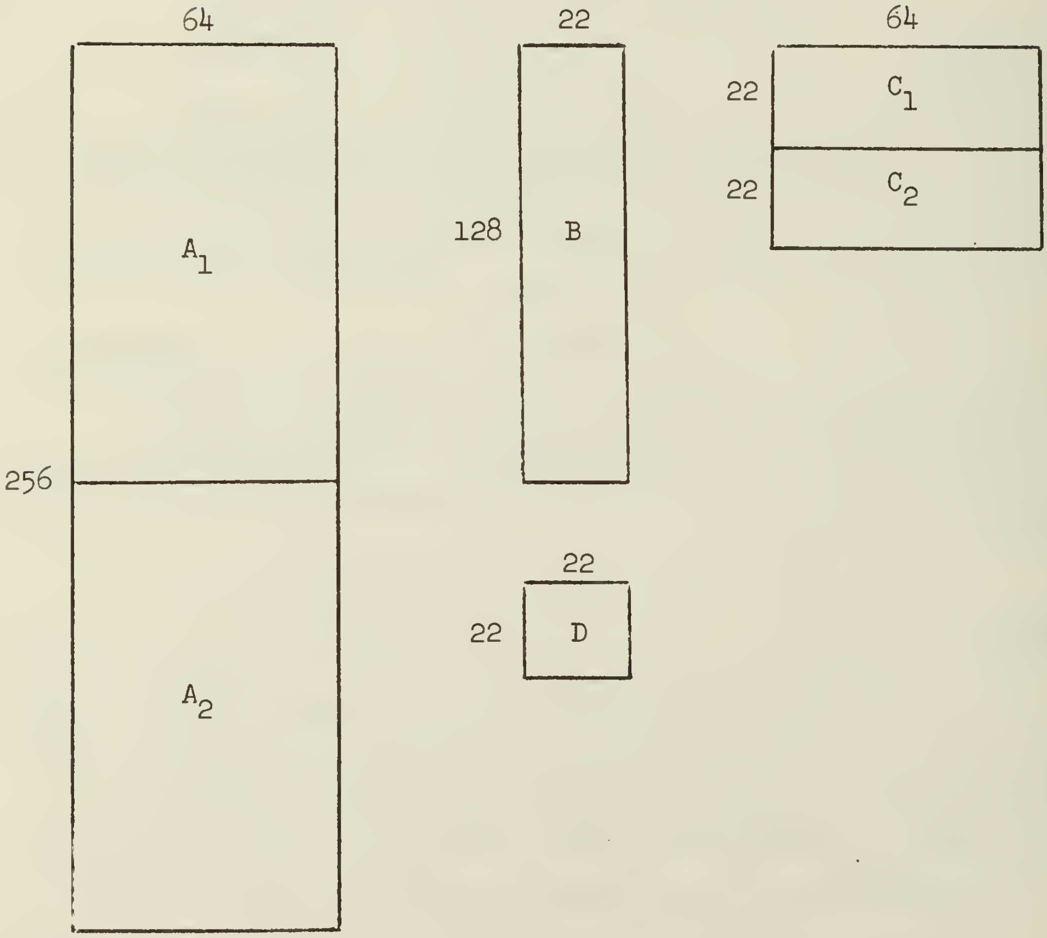

Figure 2. PE Memory Blocks

processing considerations are being examined and look somewhat promising. A preliminary language for describing geometric input and desired calculation is also being developed concurrently, with thought being given to an interactive real-time graphics display system.

2.5 Assembler

The Assembler is currently running on the B5500 and enjoys a large complement of users.

A reference manual has been written for the ILLIAC IV Assembler. It gives a preliminary specification for the basic ILLIAC IV Assembly Language, ASK version II. Included in the manual are:

1) a syntactic description of an ASK program,
2) a definition of assembly-time arithmetic,
3) a definition of assembly-time functions (pseudo-operations),
4) a definition of assembly-time utilities (control cards), and
5) examples of system control cards necessary to use ASK.

The document assumes some prior familiarity with ASK and a complete familiarity with the instruction set for ILLIAC IV as defined in the ILLIAC IV Systems Characteristics and Programming Manual.

2.6 Utility Data Processing

2.6.1 Disk Storage Allocation

A model has been developed and specified for a computer having a fixed-head disc-type of secondary memory. The purpose of this model is to allow optimization of I/O times for programs which require large amounts of input and output. The model permits this problem to be formulated as a linear programming problem. The LP formulation is relatively general in that it allows for various (computationally equivalent) reorderings of the original code as well as parameterization of memory and data transfer characteristics of the particular system being modelled.

2.6.2 Geometric Specifications

The geometry specifications processor is being revised to a preprocessor form in an attempt to allow the large number of otherwise serially performed compiler time computations to take place in parallel on the ILLIAC IV at run time. Also the power of what can be specified in the geometry specifications will increase since regular TRANQUIL arithmetic and control statements will be allowed but perhaps in a modified form.

3. APPLICATIONS

3.1 Numerical Analysis

3.1.1 Ordinary Differential Equations

Work is proceeding on an automatic integrator for ordinary differential equations for ILLIAC IV. This program will accept as input a set of ordinary differential equations and initial values and will produce an assembly code to solve them. The object is, of course, to best utilize the inherent parallelism of ILLIAC IV in solving these equations. A simple example will show some of the problems involved.

$$y_1' = a\ y_1\ y_2 + k + c \qquad y_1/y_2$$

$$y_1' = d\ y_1 + b\ y_1\ y_2 + d\ y_2$$

To solve these equations on a two processor machine, it is desirable to evaluate the first term of the first equation and the second term of the second equation together. The other terms could also be done partly in parallel. On a larger system this becomes more complex. The automatic integrator program will do the necessary analysis to obtain the maximum efficiency. Another problem is that the equations may vary greatly in the number of terms. Thus work must be shared between processors. The program is designed to work on first order systems. (It can be shown that a higher order equation can be reduced to a first order system of equations.)

The part of the program which analyzes the equations and allocates storage so as to make best use of the machine has been completed. The scanner portion which must parse the input given in the form of a statement of the equations and initial values has been written using the Translator Writing System. Development is now proceeding on the numerical methods to be used for integrating these equations. It is planned to provide some type of predictor-corrector method as well as a single-step method.

3.1.2 Multidimensional Compressible Hydrodynamics

The one dimensional program on the B5500 has been recoded to run on the 360/75 using the Calcomp plotter. So far, satisfactory results have been obtained for the constant velocity piston problem. The rest of the test problems of Hicks will be run on the 360/75 as soon as possible using the OIL, donor, and continuous rezone difference schemes. Also, the coding of these algorithms into two dimensional cylindrical coordinates with axial symmetry has been started.

Further results were obtained on the B5500 using the continuous rezone method which indicated that this method yields results as good as, or better than, OIL and donor. Various methods were tried to improve the basic algorithm of continuous rezone to give better results for the collision-of-two-shocks problem. In particular, the case in which the velocities of two adjacent cells have opposite signs was examined and several modifications tried. However, none of these were an improvement over the standard method.

3.1.3 Eigenvalue Problems

The "Triangular-Skew" method has proved to be the best storage scheme in Jacobi's method for eigenvalues of symmetric matrices. With this method of storage, the origin of the matrix is moved two locations to the right and one location down after each transformation of the matrix. After each such displacement of the matrix, the elements must be accessed each time as needed. To accomplish this, a function f calculates, at the start of the program, the PE number and location of each element as functions of the matrix indices of the element. Then a function g calculates the new PE number and location as functions of the original PE, the location, and the number of transformations done.

Take a random element (m,p) of an n x n matrix. Then

$$f(m,p) = (\text{location, PE number}) = (L_0, P_0)$$

and

$$g(f(m,p)) = g(L_0, P_0) = (L_1, P_1)$$

$$g^2(L_0, P_0) = g(L_1, P_1) = (L_2, P_2), \text{ and so on,}$$

where

$$f(m,p) = \begin{cases} (m - 1, [-n \left\lfloor \frac{p+m-2}{n} \right\rfloor + p + m - 2]), & 1 \le m \le \frac{n}{2} \\ (m - 1, \ p - \frac{n}{2}), & m = \frac{n}{2} + 1 \\ (p - \frac{n}{2}, \ m + p - n - 1), & \frac{n}{2} + 2 \le m \le n \end{cases}$$

$$g(L_s, P_s) = ([L_0 + s - (\frac{n}{2} + 1) \left\lfloor \frac{L_0 + s}{n/2 + 1} \right\rfloor], [-n \left\lfloor \frac{P_0 + 2_s}{n} \right\rfloor + P_0 + 2_s]),$$

where s denotes the number of the transformation, and $\lfloor \ \rfloor$ denotes integer arithmetic.

3.1.4 Monte Carlo Transport

During this period, a one-dimensional Monte Carlo radiation transport code, which includes scattering, was written for the B5500. The purpose for developing this program was to gain a thorough understanding of the difficulty involved in the Monte Carlo formulation. This code will be expanded to include two dimensions and then will be programmed for ILLIAC IV.

3.1.5 Polynomial Root Finding

During this quarter, work began on testing the polynomial root finding algorithm, previously developed, to determine what course of action needs to be taken when the method fails or when the polynomial is ill-conditioned. The algorithm is divided into two parts: (1) a multisectioning method, previously described, for locating simple, real roots of polynomials with real coefficients and (2) a complex integration procedure which divides a closed contour in the complex plane into sixty-four partitions and then determines the number of roots in each partition. This method is based on the theorem that:

$$S_n = \frac{1}{2\pi i} \int_C z^n \frac{f'(z)}{f(z)} \, dz = \sum_{i=1}^{v} z_i^n$$

where the z_i's ($i = 1, \ldots, v$) are the zeroes of $f(z)$ which lie in the interior of C, a closed contour in the complex plane. If $n = 0$ then S_o is merely the number of zeroes (minus the number of poles) of $f(z)$ on the interior of C. This integral can be evaluated using the trapezoidal rule since the accuracy needed is to the nearest positive integer.

The method fails when a zero is too close to the boundary of the contour C because $F(z) \to 0$ and $|f'(z) / f(z)| \to \infty$. This necessitates halting the integration and relocating the contour. Work is also being done to determine a partition algorithm which will ensure a better rate of convergence.

3.1.6 Numerical Quadrature

A program has been written in ILLIAC IV Assembly Language to implement Romberg Integration. The user supplies an error tolerance along with the required parameters and a subroutine to evaluate the function to be integrated. Efficiency is achieved by evaluating the function at a large number of points simultaneously and then picking up the values as needed to form the trapezoidal sums which are used to calculate the integral from the standard Romberg formula. If the accuracy is not achieved, additional trapezoidal sums are formed until the necessary accuracy is realized. The basic Romberg reduction is not highly efficient, but if the functions to be integrated are complex, the overall efficiency will be high due to the method of evaluating the function.

3.1.7 Mathematical Subroutines

Using codes and the ILLIAC IV document "New and Revised Mathematical Subroutines for ILLIAC IV" written by Michael Katz, the subroutines for 64-bit cosine, arc-tangent, sine, and square root were recoded, debugged, and simulated to the stated accuracy. The sine-cosine routine uses seven terms in the Chebychev expansion to obtain decimal significance equivalent to the argument. The sine-cosine scheme requires 1331 words of storage. The square root routine uses three Newton-Raphson iterations for 64-bit accuracy with fewer iterations available for less accuracy, if desired, and it requires 1321 memory locations.

Work has also begun on writing new routines, including log, arc sine, arc cosine, and others. The most common functions will also be written in double precision. An ALGOL code that converts the simulator's octal output to decimal format was written and used in debugging these routines.

The study of double precision arithmetic operations on ILLIAC IV was started during this quarter. The data format is as follows: each operand is expressed by a pair of high and low-order 64-bit, floating point numbers in such a way that the 96-bit mantissa of the operand is put in each mantissa part consecutively, and the exponent of the high-order part is that of the operand and larger than the next low-order part by 48. This data format in double length is kept throughout each arithmetic operation. In other words, data could be always a combination of two consecutive PE-memory words.

Addition and subtraction programs have been written in Assembly Language using EAD and ESB instructions, respectively. Timing is estimated at 70 clocks. For multiplication two versions have been coded. One of them is intended to obtain a result of quadruple length. Only the first two higher-order parts can be obtained by the other program. Timing for this program is estimated to be 80 clocks. Some of the above programs have been simulated and further study on multiple precision division is being considered.

Work has continued on significant digit arithmetic during this quarter. Codes for SDA multiplication and division have been simulated. Timing estimates for multiplication and division are three and six microseconds, respectively.

3.2 Linear Programming

During this quarter, the following LP algorithms have been completely defined and set up for implementation on ILLIAC IV:

1) primal,
2) dual,
3) right-hand-side parametric programming,
4) cost row parametric programming, and
5) right-hand-side and cost-row simultaneous parametric programming.

The effort of coding the algorithms is proceeding with significant progress in the tasks of refining coding specifications for portions of the minor iteration procedure and in the interfacing of the various interconnected sections of the code. The coding specifications have been expanded to include free, bounded, and fixed variables, in addition to the standard unbounded variables. This involved revision of pivot selection rules and procedures for updating the basis, right-hand side, and objective function (cost row). These revisions have been partly incorporated in the code.

The interfacing of the different sections of code is largely a problem of memory allocation, communication, and standardizing internal data formats. PE memory for buffers, work areas, and common tables has been allocated for maximum overlap of common data utilization. I/O is coordinated among subroutines so that as much I/O as possible is masked by other machine operations. Certain I/O procedures have been generalized and isolated for use by all portions of the program through common data and buffer regions.

The preprocessing of input data for ILLIAC IV has been analyzed, and an ALGOL program to prepare input data files has been written and is being debugged. This program will scale rows and columns to reduce all bounds to a common value of 1 and to minimize the size-range of coefficient values. Tables will also be prepared to allow postprocessing of the solution on the B6500.

As a first step in the process of code-testing, test matrices with known solutions have been prepared and formatted in a way acceptable to the preprocessor. These will provide a data base for code-testing during the next quarter.

Round-off errors are an ever-present problem in LP computation, so a study is presently under way to find methods of increasing the numerical stability of solutions. The methods under consideration are iterative refinement of intermediate products of the solution steps and L-U decomposition of the basis inverse.

During the next quarter, it is expected that simulation of the code on the B5500 will be commenced.

3.3 Long Codes

During this quarter, the study of stability of linear Hamiltonian systems was completed. The following is a summary of the study.

A linear Hamiltonian system with a periodic coefficient matrix may be defined by:

$$\dot{Y}(t) = J\ H(t)\ Y(t)$$

where

$$J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix},$$

$H(t) = H(t + \omega)$ is a symmetric matrix of order $2k$,
$Y(t)$ is the normalized solution of the system (Matrizer), and
$Y(0) = I$.

The above Hamiltonian system is stable for <u>all</u> time t, in the sense that the solutions $Y(t)$ do not grow with time and remain bounded as $t \to \infty$, if and only if all the eigenvalues of the monodromy matrix $Y(\omega)$ have unit modulus and all the elementary divisors are linear, i.e., if $Y(\omega)$ may be reduced to a diagonal matrix by a normal transformation.

The monodromy matrix $Y(\omega)$ of the above Hamiltonian system is defined by

$$Y(t+\omega) = Y(t)\,Y(\omega) .$$

Moreover, for the Hamiltonian system

$$\dot{X}(t) = J\,H(t)\,X(t)$$

where $X(t)$ is any fundamental matrix of the system, then it can be proved that

$$X(t) = Y(t)\,X(0) ,$$

where $Y(t)$ is the matrizer of the system and consequently $Y(t)$ is a canonical transformation which in turn can be shown to be a symplectic matrix. Symplectic matrices form a group under multiplication and the most important properties of such matrices are:

i) $Y^t JY = J$

ii) $\det(Y) = +1$.

The monodromy matrices of a system form a group under similarity transformations, and, hence, the eigenvalues of all the monodromy matrices of a given system are the same. The eigenvalues of a monodromy matrix are called the multipliers of the system and they are invariants of the system.

The above concept of stability rarely meets the requirements of any physical application. Therefore, the differential equations of motion are known only approximately, and, consequently, one needs a stability concept which allows for small changes of the equations. Such a stability concept is called Parametric stability or Strong stability.

A canonical system

$$\dot{Y}(t) = J\,H(t)\,Y(t)$$

is called parametrically stable if it is stable and if there exists an $\epsilon > 0$ such that for any real symmetric matrix $H(t)$, with piecewise continuous elements, satisfying the condition

$$\tilde{H}(t) = \tilde{H}(t + \omega) \text{ and}$$

$$\|\tilde{H}(t) - H(t)\| < \epsilon$$

(where $\|B\|$ is to be understood as max $|\beta_i(t)|$, where $\beta_i(t)$ are the eigenvalues of $B(t)$) , all solutions of the system

$$\dot{Y}(t) = J\ \tilde{H}(t)\ Y(t)$$

are bounded as $t \to \infty$.

As can be seen from the above definition, parametrically stable systems form an open set in the set of all canonical systems. Thus the conditions of parametric stability can be stated as follows.

> The system $\dot{Y} = J\ H(t)\ Y$ is parametrically stable if and only if the following three conditions are satisfied.
>
> 1) The monodromy matrix $Y(\omega)$ is diagonalizable,
>
> $D = F^{-1}\ Y(\omega)\ F = \text{diag}\ (\lambda_1, \lambda_2, \ldots, \lambda_{2k})$.
>
> 2) All the eigenvalues of $Y(\omega)$ lie on the unit circle, i.e., $\lambda_s = e^{i\omega\beta_s} = e^{i\theta_s}$.
>
> 3) Since $Y(\omega)$ is real and also symplectic, then by Liapunov-Poincaré theory, the eigenvalues of $Y(\omega)$ appear in complex conjugate pairs. Therefore, the third condition is: <u>Repeated multipliers of the unlike type do not occur on the unit circle</u>. The term "unlike type" means multipliers that leave the circumference of the unit circle in opposite directions under a complex perturbation of the coefficient matrix $H(t)$.

Condition (3) therefore can be put in any of the following forms:

a) $\lambda_{s'} \neq \bar{\lambda}_{s''}$ $\qquad (s', s'' = 1, 2, \ldots, k)$

b) $\theta_{s'} \neq -\theta_{s''}$
(assuming that the eigenvalues are arranged as $\lambda_1, \lambda_2, \ldots, \lambda_k;\ \bar{\lambda}_1, \bar{\lambda}_2, \ldots, \bar{\lambda}_k$).

c) $\sum_{s=1}^{k} \lambda_s^{g_s} \neq 1$

where, g_s are integers such that

$$g_s \geq 0, \quad \sum_{s=1}^{k} g_s = 2.$$

d) The quadratic expression (inner product), (-iJh, h) on the eigen-subspace corresponding to a repeated multiplier of multiplicity r is sign-definite, where

$$h = \sum_{v=1}^{r} \alpha_v f_v$$

and f_v are the eigenvectors of $Y(\omega)$ corresponding to the repeated multiplier.

Furthermore, the question of the "Regions of Parametric Stability of Linear Hamiltonian Systems" was studied in detail. This question arises from considering the conditions under which there exists a symmetric matrix H(t,v), piecewise continuous in t and continuous in v such that:

$$H(t,0) = H_1(t)$$

$$H(t,1) = H_2(t)$$

and such that the system

$$\dot{Y}(t) = J\,H(t,v)\,Y(t)$$

where,

$$H(t + \omega, v) = H(t, v) \qquad 0 \leq v \leq 1 ,$$

is parametrically stable for all v, $0 \leq v \leq 1$.

It can be shown that the above system is parametrically stable if $H_1(t)$ and $H_2(t)$ belong to the same stability region. A stability region is characterized by an integer n,

$$n = \left(\frac{1}{2\pi}\right)\left[\text{Arg det } Z(t)\Big|_0^\omega - \sum_{s=1}^{k} \theta_s\right]$$

where,

i) $Z(t) = \left(-iJ\, Y(t) h_v, h_\mu\right) \qquad v, \mu = 1, \ldots, k$

and $(h_1, h_2, \ldots, h_k)$ is an arbitrary basis in the subspace spanned by the eigenvectors of $Y(\omega)$ corresponding to the multipliers of the first kind $e^{i\theta_s}$, $s = 1, 2, \ldots, k$ (i.e., those eigenvalues that leave the circumference of the unit circle towards its interior under complex perturbations of $H(t,v)$).

ii) $\text{Arg det } Z(t)\Big|_0^\omega$

means the change of the argument of the determinant of Z(t) over the period (ω).

3.4 Graphics

3.4.1 Software

The basic algorithm for the hidden line elimination has been developed on the B5500. A simple, fast algorithm is essential because there are today many applications for which it is desired to view a moving object on a CRT. The old hidden line information, which is calculated once, has not been used for the subsequent computation of the hidden lines if the objects are translated or rotated about some axis. With the present methods, the computations are started over again to calculate the hidden parts for a transformed object.

The relationship between two polygons is changed whenever the rotation about the y-axis causes the plane perpendicular to the view plane to pass critical planes. The problem is how to find those critical planes for each pair of polygons. The algorithm for this has been developed. The program for a rotating object will be implemented on the B5500. This critical plane method is expected to lead to a great reduction of computation time for the hidden line elimination of a rotating object.

3.4.2 Language

The concept of the language for representing three dimensional objects has been developed. To date, users have had to give detailed data consisting of the coordinates of vertices and the description of the relationship between faces, edges, and vertices. With this language, complicated three dimensional assemblies are easily expressed by a few simple operations among some geometric operands, i.e., convex primitive solids. Each object has its own coordinate axes. The connection of primitives and the relationship of assemblies are easily expressed by the relative coordinates. Of course, the absolute coordinate expression is also possible. The overlapping of two objects in the physical world can be checked, and the adjustment can be done automatically. Furthermore, because of these features, this language will also be used in the case of the man-computer direct interaction through a cathode ray tube and a keyboard. The data structure of the language is rather complicated, but the level structure of faces and the use of convexity or concavity of solids reduce the computation time of the hidden line elimination.

3.5 Radar Data Processing

During this period, time has been devoted to reprogramming the Kalman Filter in a version in which everything is kept in spherical radar coordinates; to programming the fast fourier transform using the Cooley Tukey algorithm for both radar and seismic signal processing; and to reprogramming the time filter designation mode in radar floating point coordinates. The final report from Auerbach Corporation on the study of using ILLIAC IV for the BMD radar signal processing operations was received; however, it had to be returned for rewrite.

At the beginning of this time period, the new version of the Kalman Filter algorithm was adopted. It performs all calculations in radar coordinates, which eliminates any coordinate conversions. For this reason, it seems to be a better algorithm and has proven to be somewhat easier to code. Information on this algorithm and data generated by it were acquired from Lincoln Laboratory. An ALGOL version of the program was immediately coded and completely debugged, and is being used to create partial results to aid in debugging ILLIAC IV Assembly Language code.

The ILLIAC IV Assembly Language version of the Kalman Filter algorithm is presently being debugged on a newly available execution and timing simulator. The simulator is still unreliable to some extent and has caused much slowdown in progress. Some portions of the assembly code are completely debugged, and some preliminary timings have been made. Debugging and timing is expected to be completed by the end of the next time period.

The FFT (Fast Fourier Transform) algorithm has been programmed in ALGOL for the B5500 and in ILLIAC IV Assembly Language ASK as a general subroutine which can be used for both radar and seismic signal processing. The version programmed in ASK is presently being debugged by using the ILLIAC IV simulator.

3.6 Seismic Signal Processing

A series of programs have been written in B5500 ALGOL to provide data for later use in testing and debugging ILLIAC IV programs. This series of programs is functioning and is now being written in ILLIAC IV Assembly Language. The programs are:

1) Auto Correlation: used to generate an auto correlation function of both individual and summed seismic traces. Such functions are used in deconvolution to determine filter weights, in the determination of the power spectrum of the trace, to determine the presence of multiple reflections, for wavelet compression, and for reverberation minimization.

2) Cross-correlation: used in filter applications and in establishing amplitude spectra of periodic signals.

3) Convolution: filter operators derived through the use of the auto correlation function are convolved with the signal trace resulting in wavelet compression and reverberation minimization.

4) Sine and Cosine Wave Generators: used to obtain sine and cosine transforms and Fourier amplitude and phase spectra.

3.7 Statistical Packages

During this quarter, work was begun on a rudimentary scanner for the input language for the ILLIAC IV statistical system. This language will, in general, use blanks as delimiters and will limit the use of other special characters. It will derive its information content primarily from positional interrelation of key words. This type of input will allow an input program to resemble more closely a natural language string. The effect of this will not be large in the limited area of statistics, but this will allow further development of an information retrieval language with a much broader purpose and which will be at least syntactically compatible with the base statistical language. The ultimate aim would be to provide almost natural language instruction to a data retrieval system which would operate in conjunction with a file maintenance system to generate input for the statistical system.

The second area of study for this quarter was data management systems. In particular, the use of DL/1 in conjunction with the Rockland State Hospital (Orangeburg, New York) mental patient file was observed.

The specifications of the original DM/1 system were also studied. Data management and its file maintenance currently represent one of the largest problem areas in the computer field. Since it is expected that large scale data bases will be used in connection with the ILLIAC IV statistical system, the feasibility of a data management system for ILLIAC IV is being studied.

3.8 Education

Again this quarter two courses were organized and offered by the ILLIAC IV Education area of the Project. The CS 491-D course, which covered the hardware, languages, and applications of the ILLIAC IV, and a course for new ILLIAC IV staff which covered the ALGOL language and methods of running programs on the B5500 were offered. A new development in this area was that the CS 491-D course was offered university-wide, and almost half of the students enrolled were non-ILLIAC IV staff.

A project has been undertaken to formulate the material covered in the CS 491-D class into a document which will be made available to people interested in learning about the ILLIAC IV project.

In the last Quarterly Progress Report, the initiation of the attempt to collect and categorize all the programs written for the ILLIAC IV machine was mentioned. This, along with the effort to compose another TRANQUIL manual, is continuing.

THESES

Beals, Alan J. "The Generation of a Deterministic Parsing Algorithm." Master's thesis, DCS Report No. 304. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

Budnik, Paul P. "TRANQUIL Arithmetic." Master's thesis, DCS Report No. 296. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

Muraoka, Yoichi. "Storage Allocation Algorithms in the TRANQUIL Compiler." Master's thesis, DCS Report No. 297. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

Trout, H. Robert G. "A BNF Like Language for the Description of Syntax Directed Compilers." Master's thesis, DCS Report No. 300. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

Tummelson, Jay M. "An Algorithm for Delay Checking Computer Designs." Master's thesis, DCS Report No. 301. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

Wilhelmson, Robert. "Control Statement Syntax and Semantics of a Language for Parallel Processors." Master's thesis, DCS Report No. 298. Urbana, Illinois: Department of Computer Science, University of Illinois, 1969.

DOCUMENTS

Katz, Michael L. "New and Revised Mathematical Subroutines for ILLIAC IV." ILLIAC IV Doc. No. 210, DCS File No. 787, (January 27, 1969).

Lawrie, Duncan H. "Glypnir Operation Manual." ILLIAC IV Doc. No. 213, DCS File No. 792, (March 10, 1969).

McCarthy, Thomas. "Solving a Set of Ordinary Differential Equations Involving a Large Matrix." ILLIAC IV Doc. No. 209, DCS File No. 786, (January 27, 1969).

Northcote, Robert S. "Software Development for the Array Computer ILLIAC IV." ILLIAC IV Doc. No. 214, DCS Report No. 313, (March 12, 1969).

Westlund, G. A., et al. "A Description of the ILLIAC IV Operating System." ILLIAC IV Doc. No. 212, DCS File No. 791, (March 10, 1969).

Yasui, Toshio, and Winje, Gilbert L. "Significant Digit Arithmetic on ILLIAC IV." ILLIAC IV Doc. No. 211, DCS File No. 789, (February 18, 1969).

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Department of Computer Science University of Illinois Urbana, Illinois 61801	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE
ILLIAC IV QUARTERLY PROGRESS REPORT
January, February, March 1969

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Jan. - March, 1969 - Progress Report of the ILLIAC IV Project

5. AUTHOR(S) *(First name, middle initial, last name)*

6. REPORT DATE May 1, 1969	7a. TOTAL NO. OF PAGES 40	7b. NO. OF REFS 12
8a. CONTRACT OR GRANT NO. 46-26-15-305 b. PROJECT NO. USAF 30(602)-4144 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) ILLIAC IV Document No. 219 DCS Report No. 339 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* RADC TR	

10. DISTRIBUTION STATEMENT

Qualified requesters may obtain copies of this report from DCS.

11. SUPPLEMENTARY NOTES NONE	12. SPONSORING MILITARY ACTIVITY Rome Air Development Center Griffis Air Force Base Rome, New York 13440

13. ABSTRACT

See the Report Summary on Page 1 within the Report itself.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
PE Logic Simulator						
PE Diagnostics Generation						
CU Logic Simulator System						
CU Diagnostics Generation						
Design Automation						
Operating System						
Translator Writing System						
Compilers and Translators						
Assembler						
Utility Data Processing						
Numerical Analysis						
Linear Programming						
Long Codes						
Graphics						
Radar Data Processing						
Seismic Signal Processing						
Statistical Packages						
Education						

UNCLASSIFIED
Security Classification